EXHIBIT C

EXPERT REPORT OF TERENCE PARR, PH.D.
REGARDING THE NON-INFRINGEMENT OF U.S. PATENT NO. 6,061,520

Appendix A

II. PATTERN MATCHING IS NOT SIMULATING EXECUTION

A-16. I developed a simple test to show that simulating execution is not the same as pattern matching.

A-17. My test consisted of modifying the static array initialization bytecode in a Java class file to inject what is functionally a "noop" into the bytecode stream. I then ran the dx tool on the modified Java class file to see if the parseNewarray method would identify the modified statically initialized array bytecode.

A-18. For this test, I compiled the following source code written in the Java programming language:

```
public class T {
        public static int[] a = {5,4,3,2,1};
}
```

A-19. After compiling this class using the standard Java compiler (i.e., into a file named "T.class"), I was able to use a library called ASM to alter the bytecode within the clinit method of the T.class file. More specifically, I added a redundant array element initialization right after I saw the newarray bytecode. This redundant array initialization would be equivalent to the following source code statement:

```
a[0] = 0;
```

A-20. This extra element results in an extra element in the static array initialization bytecode pattern, i.e., the following bytecode sequence:

```
3:   dup
4:   iconst_0
5:   iconst_0
6:   iastore
```

A-21. When I ran the dx tool on the original, unmodified T.class file, the dx tool's pattern matching mechanism identified the statically initialized array and generated the fill-array-data instruction.

A-22. In contrast, when I ran the dx tool on the modified T.class file, the dx tool's pattern matching mechanism failed to recognize the modified class file as fitting the static array initialization and does NOT generate a new fill-array-data instruction.